Wireshark Notes

- Use off **logical operators are allowed** (i.e.: **&&** -> **and**, **||**-> **or**, etc.)
- **Curl** -> tool to transfer data from Host to a Server
  - Use command "**man curl**" for details
  - **curl -X <REQUEST TYPE> google.com** -> for different types of request(i.e.: POST, PUT etc.).
  - **curl -X POST http://google.com**
  - curl makes GET request by default
- **Net cat** -> Use net cat for direct message communication between devices to check connections.

Retrieve rows using IP:

Compare source IP address logically:

ip.src == 192.168.1.1

Compare destination IP address logically:

ip.des == 172.148.1.9

Search for protocols:

DNS, HTTP, HTTPS, TLS, SSL etc.

- Search in search block provided in Wireshark

Use of sub functions:

- Sub functions are provided for various protocols. E.g.:
  - TCP (**tcp.port == 443**)
  - UDP (**udp.port == 80**)
  - Retrieving rows with matching ports

Contains keyword:

See if TCP or UDP contains some specific information.

E.g.: **tcp contains GET** -> check if request/response contains a GET request

- http.request.method -> logically check for specific request methods. E.g.:
    - **http.request.method == GET**
    - **http.request.method == POST**
- On network request: **Right Click > Follow > TCP Stream (View Plain Text)**

NetCat:

- Use net cat for direct message communication between devices to check connections.
    - It can also be used to send unencrypted messages from the source device to destination.
    - For testing:
        - Listener (i.e.: destination): **nc -l -p 9999**
        - -l -> used for listening
        - -p -> port
        - Host (i.e.: Sender): nc <IP> <PORT>
        - **nc 192.168.1.5 9999**

Retrieve FTP data:

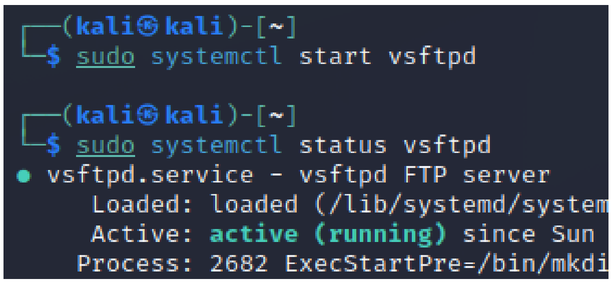
Testing:

Setup FTP server in Kali for testing.

Setup FTP file sharing in Kali for testing.

Steps:

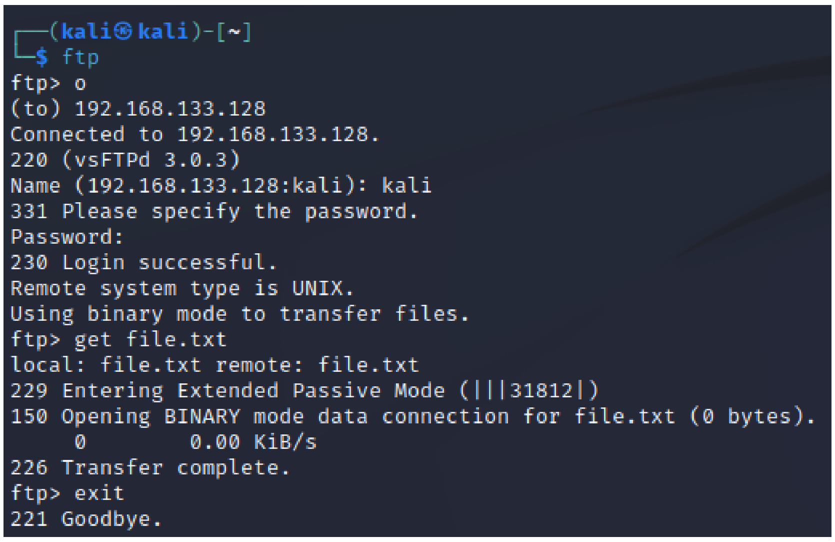
1. sudo apt update
2. sudo apt install vsftpd
3. sudo systemctl start vsftpd
4. sudo systemctl status vsftpd (Check if status is active)



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start vsftpd

┌──(kali㉿kali)-[~]
└─$ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/lib/systemd/system
     Active: active (running) since Sun
    Process: 2682 ExecStartPre=/bin/mkdi
```

5. create a file and fill in your data
6. start terminal

7. 

```
┌──(kali㉿kali)-[~]
└─$ ftp
ftp> o
(to) 192.168.133.128
Connected to 192.168.133.128.
220 (vsFTPd 3.0.3)
Name (192.168.133.128:kali): kali
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get file.txt
local: file.txt remote: file.txt
229 Entering Extended Passive Mode (|||31812|)
150 Opening BINARY mode data connection for file.txt (0 bytes).
     0        0.00 KiB/s
226 Transfer complete.
ftp> exit
221 Goodbye.
```

8. Data transfer is now completed and ready to be viewed in Wireshark

Retrieving data:

- FTP file transfer
- Search ftp-data to get file data(follow > tcp stream)
- View Login info from HTTP req > HTTP Protocol, POST request, HTTP form URL Encoded

Images:

Testing:

Open a HTTP website with images.

Retrieving data:

- Search(in Wireshark) jpeg, png, etc to find images available.
- Stop traffic capture then go to: File > Export Objects > HTTP
    - **Text Filter: png/jpeg etc**
    - Select and **save** or **save all**
    - And view the images in your desktop.

Plain Text and network protocols:

- Some network protocols do not use encryption. Such protocols are called clear text (or plain text) protocols.
- All the data is visible to the naked eye, including passwords.
- Anybody who is in position to see the communication (e.g., man in the middle) can ultimately see everything.
- Underlying is some of the protocols with port numbers that generally do not encrypt the data flowing through them

Port	**Service**	**Name**
TCP/20, TCP/21	FTP	File Transfer Protocol
TCP/23	Telnet	Teletype Network Protocol
TCP/25	SMTP	Simple Mail Transfer Protocol
TCP/80	HTTP	Hyper Text Transfer Protocol
TCP/110	POP3	Post Office Protocol
TCP/143	IMAP4	Internet Message Access Protocol
UDP/161, UDP162	SNMP	Simple Network Management Protocol
TCP/389	LDAP	Lightweight Directory Access Protocol
TCP/1080	SOCKS	SOCKetS Proxy Protocol
TCP/1433	MSSQL	Microsoft SQL Database
TCP/5222	XMPP	Extensible Message and Pressure Protocol
TCP/5432	PostgreSQL	PostgreSQL Database
TCP/6667	IRC	Internet Relay Chat